# **House Property Sales Analysis**

**Team Members – Group 2**

1. Aamir Qutubuddin
2. Anupama Nambanoor Kunnath
3. Ayara Dixson
4. Santanu Poddar

**Abstract**

The purpose of the project is to analyze the house property sales over the time.

Apart from that, Anomaly Detection and Forecasting were performed on the Sales Price of the house using different SKLearn and AWS Models.

As part of the course project, we will be executing the following tasks using the dataset below:

<https://www.kaggle.com/code/aurygabrielawestcott/house-property-sales-analysis/data?select=raw_sales.csv>

1. Regression Task – using SKlearn Model
   * Linear Regression
   * Decision Tree Regressor
2. Regression Task- using AWS Model
   * XGBoost
3. Anomaly Detection Task – using SKlearn Model

* Isolation Forest
* Local Outlier Factor

1. Anomaly Detection Task – using AWS Model

* Random Cut Forest

1. Forecasting Task - using SKlearn Model

* Support Vector Machine
* Decision Tree
* Random Forest

**Introduction**

The retail industry now heavily relies on data analytics tools to better estimate the prices of different properties. Work on this project idea deals with analyzing the sales of house properties in a city in Australia.

**Dataset**: [The House Property Sales dataset](http://www.kaggle.com/datasets/htagholdings/property-sales) on Kaggle contains a file named ‘raw\_sales.csv.’ It includes the following variables:

<https://www.kaggle.com/code/aurygabrielawestcott/house-property-sales-analysis/data?select=raw_sales.csv>

Dataset has 5 columns as below:

* **Datesold**: The date when an owner sold the house to a buyer.
* **Postcode**: 4-digit postcode of the suburb where the owner sold the property.
* **Price**: Price for which the owner sold the property.
* **Property Type**: Property type i.e., house or unit
* **Bedrooms**: Number of bedrooms.

**Preliminary Analysis**

Below are the features and target data used to perform the Regression Task using SKLearn models and AWS model.

**Features**

* **Datesold**: The date when an owner sold the house to a buyer.
* **Postcode**: 4-digit postcode of the suburb where the owner sold the property.
* **Property Type**: Property type i.e., house or unit
* **Bedrooms**: Number of bedrooms.

**Target**

* **Price**: Price for which the owner sold the property.

**Data Preprocessing**

**Pipeline for Data Preprocessing**

A pipeline allows us to combine all the steps into one single object that can be easily applied onto different parts of the data. In this project, we have built a processing pipeline that includes

* Train/Test split with stratified split
* Standardization
* Imputation
* One Hot Encoder

We have separated the preprocessing of numeric and class columns in two different pipelines and combine them using ColumnTransformer.

**Standardizing Numeric Columns**

In standardization, we transform the values in the column so that they have a mean of 0 and standard deviation of 1

In SKLearn, we can use **sklearn.preprocessing.StandardScaler**. The **fit\_transform()** function will do everything in one step.

StandardScaler cannot work on class attributes and will report errors if your give it a DataFrame that has class attributes. We need to filter out such columns, which can be done with **DataFrame.select\_dtypes('number')**

### **Imputation**

A preferred way handling the missing data is to do **impute**. This means to replace missing values with a certain value.

If the column with missing value is a class column, we can simply replace the missing values with a missing class, for example, 'missing'. This can be done quite simple with **DataFrame.fillna()**. we need to save the new data in a new variable. These new 'missing' values will be dealt with in the next task.

If the missing value is a numeric field, we use the median value to replace the missing cells.

**Modeling and Results**

We have used the following models using the dataset:

<https://www.kaggle.com/code/aurygabrielawestcott/house-property-sales-analysis/data?select=raw_sales.csv>

1) Regression Task – using SKlearn Model

* + Linear Regression
  + Decision Tree Regressor

2) Regression Task - using AWS Model

* + XGBoost

3) Anomaly Detection Task – using SKlearn Model

* Isolation Forest
* Local Outlier Factor

4) Anomaly Detection Task – using AWS Model

* Random Cut Forest

5) Forecasting Task - using SKlearn Model

* Support Vector Machine
* Decision Tree
* Random Forest

**Results**

**Regression**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Linear Regression** | **Decision Tree Regressor** | **AWS XGBoost** |
| **Accuracy Score - Training** | 0.3481287263526407 | 0.5763631867718085 |  |
| **Accuracy Score - Testing** | 0.3396389753087887 |  | 0.00% |

**Anomaly Detection**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Isolation Forest** | **Local Outlier Factor** | **AWS - Random Cut Forest** |
| **Anomaly Percentage** | 2.04 % | 4.65 % | 1.12 % |

**Forecasting**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Support Vector Machine** | **Decision Tree** | **Random Forest** |
| **Score** | -0.2821318253673877 | 0.10207247650375784 | 0.050569616071615364 |

**Hyperparameters**

The following hyperparameters have been used for Forecasting task to finetune the different models.

**Support Vector Machine**

param\_grid = [{

'C': [0.01, 0.1, 1, 10, 100],

'kernel' : ['rbf'],

'gamma' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]

}]

**Decision Tree**

param\_grid = [{

'max\_depth': [3,4,5,6],

'max\_features' : [4],

'min\_samples\_split' : [2, 10, 20, 30, 40],

'min\_samples\_leaf' : [1, 10, 20, 30, 40]

}]

**Random Forest**

param\_grid = [{

'n\_estimators' : [5, 10, 20, 50],

'max\_depth': [3,4,5],

'max\_features' : [4],

'min\_samples\_split' : [2, 10, 20, 30, 40],

'min\_samples\_leaf' : [1, 10, 20, 30, 40]

}]

**Which model performs the best**

* For Regression Task, the Decision Tree Regressor model gave the best optimal result of the dataset used.
* For Anomaly Detection Task, the Local Outlier Factor model gave the best optimal result of the dataset used.
* For Forecasting Task, the Decision Tree model gave the best optimal result of the dataset used.

**Conclusion**

We had to use a slightly different approach for anomaly detection in addition to what we learned from the tutorials due to the specialty of the dataset. Since the dataset has different zip codes, it would give false anomalies in house price due to possibility of having two consecutive rows being two different cities.

Finding the correct data that could suffice for all the models was a predicament. I faced issues with regression, such as the SVM model taking forever to run, so I had to stop it forcibly, and Decision Tress threw a warning with the approach followed by the class tutorials. It was challenging to find solutions on the internet. So, we included those regression models which provided an optimal result. Overall, it was a great experience working on a group project to analyze house property sales over time.

Finding the right accuracy score for regression task using AWS XGBoost model was one of the challenges for us using the dataset.

Determining what window size you want to use will decide the type of answers you will receive from your dataset. I had no difficulty once I decided my window size and tan my code for the forecasting.

It was an absolutely good experience in researching on adopting best practices based on varying datasets. It made us deeply understand the actual purpose of each machine learning step.